Implementation of Pin Point Landing Vision Components in an FPGA System

Arin Morfopolous, Brandon Metz, Carlos Villalpando, Larry Matthies, Navid Serrano
Arin@jpl.nasa.gov, Brandon.metz@jpl.nasa.gov, Carlos.Villalpando@jpl.nasa.gov, Larry.H.Matthies@jpl.nasa.gov
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109
Navid.Serrano@adventbt.com

Abstract— Pin-point landing is required to enable missions to land close, typically within 10 meters, to scientifically important targets in generally hazardous terrain. In Pin Point Landing both high accuracy and high speed estimation of position and orientation is needed to provide input to the control system to safely choose and navigate to a safe landing site. A proposed algorithm called VISion aided Inertial NAVigation (VISINAV) has shown that the accuracy requirements can be met. [2][3] VISINAV was shown in software only, and was expected to use FPGA enhancements in the future to improve the computational speed needed for pin point landing during Entry Descent and Landing (EDL). Homography, feature detection and spatial correlation are computationally intensive parts of VISINAV. Homography aligns the map image with the descent image so that small correlation windows can be used, and feature detection provides regions that spatial correlation can track from frame to frame in order to estimate vehicle motion. On MER the image Homography, Feature Detection and Correlation would take approximately 650ms tracking 75 features between frames. We implemented Homography, Feature detection and Correlation on a Virtex 4 LX160 FPGA to run in under 25ms while tracking 500 features to improve algorithm reliability and throughput. [1] [2]

TABLE OF CONTENTS

1. INTRODUCTION....1
2. THE VISINAV SYSTEM....2
3. FPGA SYSTEM OVERVIEW....2
4. MEMORY INTERFACE....3
5. HOMOGRAPHY....3
6. FEATURE DETECTION....4
7. NORMALIZED CROSS CORRELATOR....6
8. FUTURE AND RELATED WORK....6
9. RESULTS....7
10. SUMMARY....7
ACKNOWLEDGEMENTS....8
REFERENCES....8
BIOGRAPHY....9

1. INTRODUCTION

NASA's roadmap for solar system exploration includes missions to the Moon, Mars, Europa, Titan, comets and asteroids, which requires an accurate estimation of position to enable safe landing near the desired science targets.[1]

Previous robotic lander missions have used a combination of inertial measurements from accelerometers and gyroscopes and velocity measurements from Doppler radar. The resulting landing ellipse is quite large because of uncertainty in the initial position at the start of EDL and the accumulation of measurement error during integration. For Mars, the landing ellipse has been on the order of 100km, and on the Moon the landing ellipse has been on the order of 1km. Absolute measurement of position via GPS or similar satellite arrays will not be available near non-Terrestrial solar system bodies for the foreseeable future.

Camera measurements can reduce the uncertainty in touchdown position. Features detected from on-board descent imagery can be matched against prior images taken from orbit (Mapped Landmarks) to bound the absolute uncertainty, and features can be detected and tracked frame to frame during landing to reduce relative uncertainty (Feature Tracking) for position, velocity and attitude.

These measurements would be used in addition to the standard array of navigation tools, and JPL's implementation of such a system is called VISion aided Inertial NAVigation, or VISINAV. [2]

In September of 2007 JPL showed that VISINAV could achieve calculated position errors under 10cm and velocity errors under 20cm/sec. Tests were performed using parachute drops, sounding rocket launches and prior planetary landing sequences to show that the VISINAV system was robust to a large range of altitudes, attitude dynamics, lighting conditions and scene appearance changes. [3]

VISINAV was implemented in C++, and timing experiments with the sounding rocket data set showed that more than 50 mapped landmarks could be generated in 2 seconds and that the persistent feature tracker could track more than 70 features in 100 ms on a 400Mhz R12000 SGI O2. At this rate the algorithm would be fast enough to support real-time landmark mapping and feature tracking during descent and landing, but a flight CPU might be significantly slower than the 400MHz R12000 and thus would not be able to hit the timing requirements desired[2]. Additionally, 100% of the CPU may not be available for VISINAV to use during EDL. Improvements were desired

[1] 978-1-4244-7351-9/11/$26.00 ©2011 IEEE.
[2] IEEEAC paper #1157, Version 1, Updated September 1, 2010

to offload the CPU of this computationally intensive task by implementing key components in an on-board FPGA. By shifting the heavy processing to the FPGA, we can also improve the robustness and precision of VISINAV by processing many more than 50 mapped landmarks and 70 features.

The implementation of VISINAV components within an FPGA are the focus of this paper.

2. THE VISINAV SYSTEM

The computationally intensive portions of the VISINAV algorithm lie in

a) Feature detection

b) Image scaling and transformation, or Homography

c) Feature matching via a 2D spatial correlator.

d) Feature matching via a Fast Fourier Transform (FFT).

e) Kalman filter state updates.

Components (a), (b) and (c) were chosen for implementation in an FPGA.

Feature matching via a Fast Fourier transform has not yet been included because in the VISINAV algorithm it is only done once during the Mapped Landmark phase, regardless of the number of features tracked. Nonetheless it is expected to be included during future development.

Kalman filter state updates was not included for FPGA implementation because it did not take a significant portion of the total computation time and because it was less well suited to a parallel or pipelined implementation.

The VISINAV system itself is extensively described in previous literature[4][5][6]. The only details of the VISINAV system described here will be the behavior of the components that were chosen for implementation in an FPGA.

This paper will present an FPGA architecture that includes implementations of image Homography using bilinear interpolation, a feature detector using a Harris operator, and a 2D spatial correlator using a Normalized Cross Correlator (NCC), as well as the data flow for each module.

3. FPGA SYSTEM OVERVIEW

The development FPGA board is an Alpha Data ADM-XRC4 containing a Virtex 4 LX160, a PCI bus interface and 6 independent ZBT SSRAM 4MB banks. The memory and PCI bus interfaces are common across this design and a related JPL project [7] to handle rover image processing for navigation.

For the flight system we expect that we would be using a Rad Hard by design Virtex 5 FX130T (part #XQR5VFX130) and a single large bank of SDRAM, on the order of 512MB to 4GB. The FPGA design supports the expected transition to this flight FPGA part and SDRAM interface.

The key design considerations for the FPGA modules were maximizing speed and minimizing FPGA resource utilization. FPGA resources are finite and the demands on resources will only grow as the flight date approaches and more capabilities are added.

Each FPGA module was designed to be fully pipelined so that the speed will be constrained only by the speed that imagery can be loaded into the FPGA and by the clock speed of the FPGA.

Each component- Homography, Feature detection and Normalized Cross Correlator (NCC)-is called an engine, and is started by the CPU by register reads and writes over the PCI bus. Once running, each engine runs to completion and notifies the CPU via a PCI interrupt of its completion. Interactions between CPU and FPGA are kept to a minimum to optimize CPU idle time (and thus free for other tasks) and PCI bus idle time.

Expected design conditions:

a) 512x512 input descent imagery, grayscale, 8 bit.

b) 2048x2048 input orbital map image.

c) Up to 500 features detected in a single descent image.

d) Up to 500 features used during Persistent Feature Tracking.

e) 15x15 Normalized Cross Correlator template size to be used for matching.

f) Input imagery will be stored in on board memory before use. Each engine has sole control over the imagery while it is running.

g) Minimum performance of 1Hz for VISINAV with the improvements in performance from the FPGA.

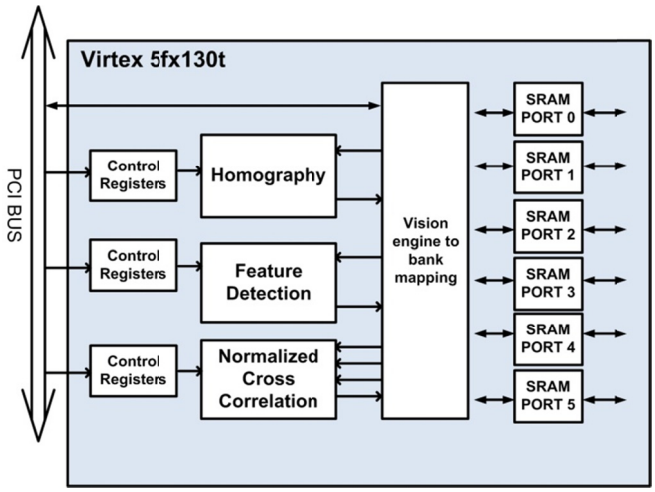
Note that 500 features tracked are far greater than in the standard software-only version of VISINAV. Both speed and performance are expected to improve as a result of the FPGA acceleration.

Each FPGA block runs independently, and in the VISINAV algorithm only one FPGA block will need to run at a time. There is no advantage to running Homography and Feature detection simultaneously, or Feature Detection and Correlation simultaneously, because each subsequent algorithm depends on the completion of the prior.

4. MEMORY INTERFACE

There are essentially 4 FPGA elements that can access on board RAM: PCI bus on behalf of the CPU, Homography, Harris Detector, and the Normalized Cross Correlator.

The imagery is separated into different on-board memory banks in order to maximize throughput, so that input data and output data are all independent accesses to memory.



1. Data flow for each engine to and from on board memory. The bank mapping is handled by an internal state machine dependent on PCI bus requests and FGPA engine states.

Each engine has different memory use profiles:

The PCI bus will access a single bank at a time to write in a new image or to read out processed data.

The Homography will read a single raw descent image from memory and write a single rectified image to memory.

The Feature detector will read from the raw descent image or from the rectified image (both inputs will be used at different points in the VISINAV algorithm) and will write feature coordinates out.

The Normalized Cross Correlator will read templates from the rectified descent image, read coordinates for features within the map window, read sub-windows from the map image at the coordinates of features detected, and write coordinates of the best match between templates and sub-windows out.

There are 6 independent banks of ZBT SSRAM on our development board. In each case listed above there are sufficient banks to allow a module to have an independent SRAM bank for each read or write operation. This simplifies design, but for the flight mission there will likely only be one large bank of SDRAM, so in that case an arbiter would be required to share access of the single bank across multiple read/write agents. This arbiter/agent design is currently under development at JPL. In the current design any module that is running has exclusive control the memory banks it uses until it completes.

Each module was written to be a stand-in replacement for its equivalent software function. Unfortunately the implementation is slightly different for the FPGA so that the results are not identical, bit for bit, between the software and FPGA versions of Feature Tracking and Normalized Cross Correlation. Substantial testing with different data sets will be required to validate that the results are equally stable and equally valid.

5. HOMOGRAPHY

2D spatial correlation is not scale or rotation invariant, so any images being matched must first be scaled and rotated properly. This applies to both the Mapped Landmark and the Feature Tracking algorithms. For the Mapped Landmark algorithm sub-windows around a feature in the descent image will be scaled and rotated to align with the map image. For Feature Tracking the sub-window around a feature in the descent image will be scaled and rotated to align with the prior descent image. Homography can run across the entire image in one pass, but we rectify the coordinates of the features found separately in software.

The descent image is loaded into an SRAM bank, then a 3x3 transformation matrix is loaded into the FPGA registers and then the Homography engine is started.

The Homography engine loads the descent image from the SRAM and remaps it on the fly, writing the rectified output pixels out to a separate SRAM bank.

The transformation between output image coordinates and input coordinates are defined by:

$$x2 = (h1*x1 + h2*y1 + h3) / (h7*x1 + h8*y1 + h9) \quad (1)$$

$$y2 = (h4*x1 + h5*y1 + h6) / (h7*x1 + h8*y1 + h9) \quad (2)$$

Where h1 to h9 are the elements of the transformation matrix, and x2,y2 are the coordinates in the unrectified descent image. Input values x1 and y1 are the coordinates in the rectified descent image. [8][9]

The computation is fully pipelined, so on each new clock a new coordinate is computed.

x2 and y2 are referenced in the input image to read the 4 neighboring pixels surrounding the floating point address (x2,y2).

These neighboring unrectified input pixels Pa, Pb, Pc and Pd are weighted to produce the rectified output pixel Pr at (x1,y1) in the output image.

$$P_r = \frac{(P_a(1-W_x)+P_bW_x)(1-W_y)+(P_c(1-W_x)+P_dW_x)W_y)}{256} \quad (3)$$

Where W_x is the weight along the x-axis and W_y is the weight along the y axis, which corresponds with the fractional portion of the (x2,y2) coordinates. Figure 2 graphically illustrates Equation 1. The dark dot represents the projected coordinate (x2,y2) of the rectified pixel into the raw image frame.

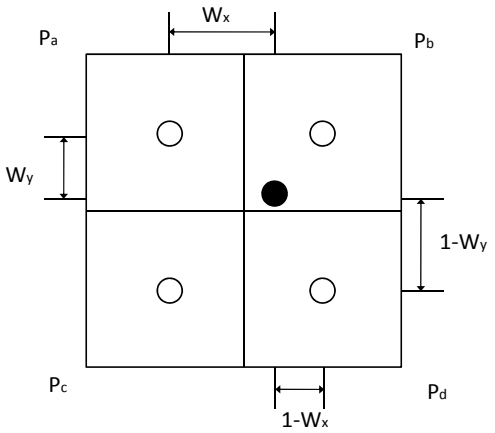


Figure 2: Homography bilinear interpolation example

Note that the equation (3) and method of bilinear interpolation is identical with that used in Rectification in JPL's Rover Navigation FPGA design [7]. The method of determining (x2,y2) and thus W_x and W_y are of course different.

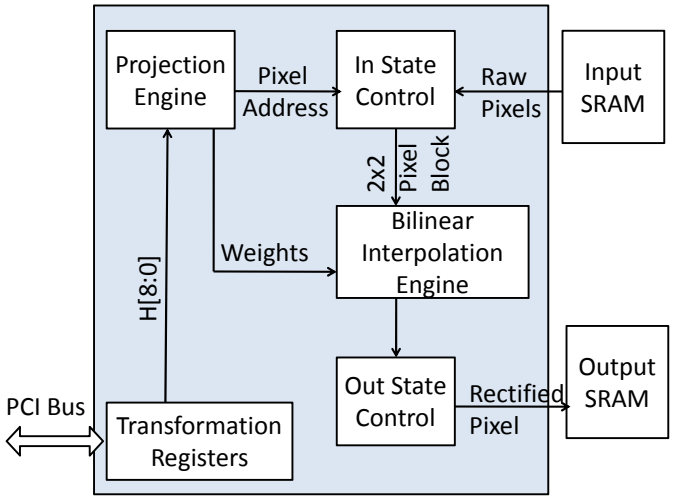


Figure 3. Homography Block Diagram

Figure 3 shows the data flow to accomplish the equations (1) (2) and (3). The Projection Engine creates the (x2,y2) floating point coordinate and translates that into an integer Pixel Address and a fractional W_x and W_y weight.

In State Control handles the interface with the SRAM to read descent image data. It is synchronized with the Projection Engine block to make sure the Weights and the 2x2 Pixel Block arrive together at the Bilinear Interpolation Engine.

Out State Control handles the output to SRAM of rectified pixels and keeps track of when all the output pixels are completed.

6. FEATURE DETECTION

The VISINAV algorithm uses a Harris Detector to locate features which can be matched to the map image or to prior descent images. Harris was chosen because it is efficient and because it generally extracts corners in textured areas of images which are well suited to correlation-based tracking.

The Harris detector is fundamentally a corner detector, finding areas with a large derivative in x and y. It functions by considering a local window in the image and determining the average changes of image intensity that result from shifting the window by a small amount in any direction (x,y)[10][11]. The change, E, for the small shift (x,y) can be concisely written as:

$$\mathrm{E}(\mathrm{x},\mathrm{y}) = (\mathrm{x},\mathrm{y})\mathrm{M}(\mathrm{x},\mathrm{y})^{\mathrm{T}} \quad (4)$$

Where:

$$M = \begin{bmatrix} AC \\ CB \end{bmatrix}$$
$$A = \partial I/\partial x * \partial I/\partial x$$
$$B = \partial I/\partial x * \partial I/\partial y \quad (5)$$
$$C = \partial I/\partial y * \partial I/\partial y$$

The corner response function is given as:

$$F = Det(M) - k * Trace^2(M) \quad (6)$$

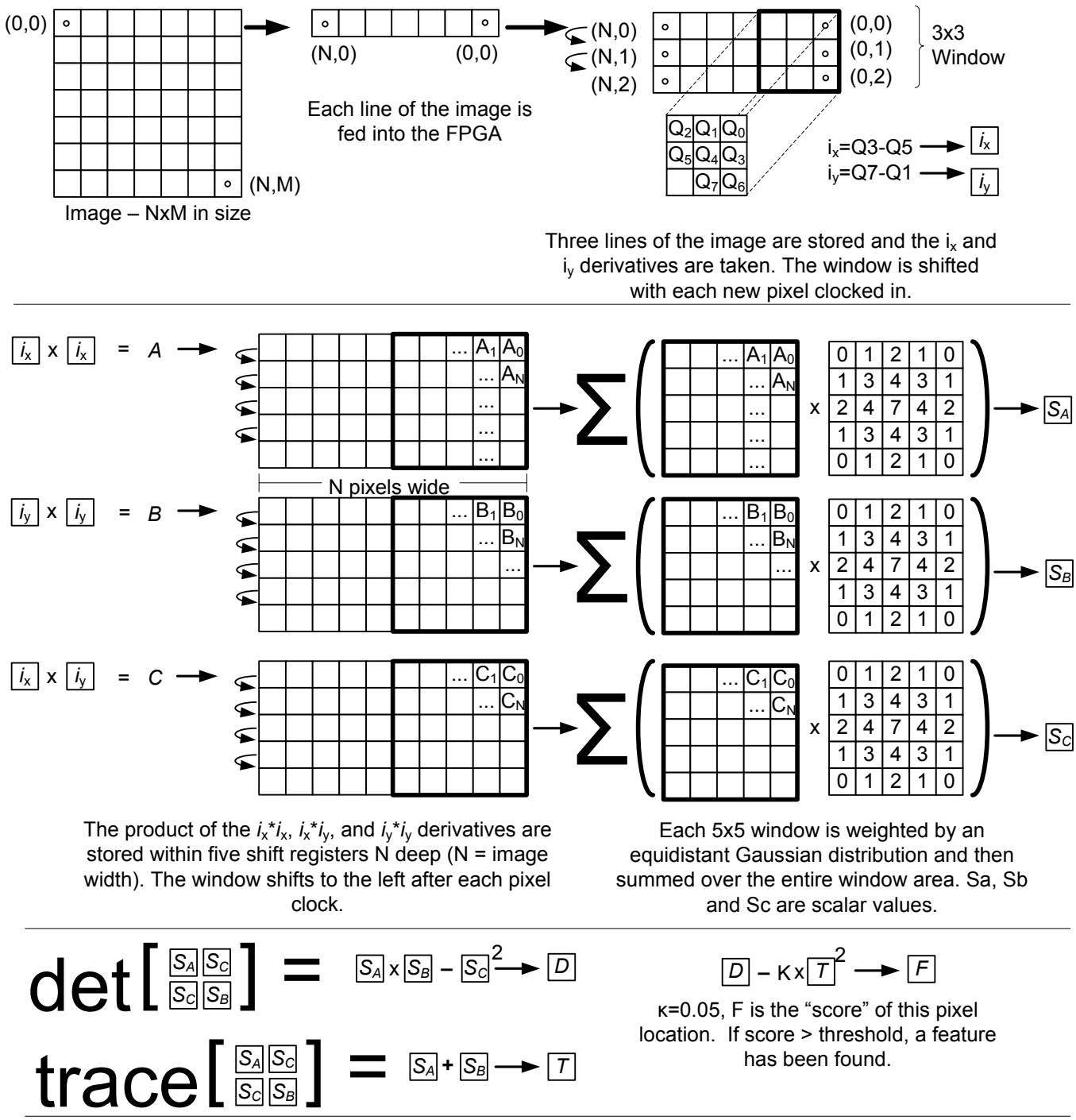


Figure 4. Harris Detector Functional Diagram

The diagram above shows each main phase of the pipeline starting with the insertion of the image, pixel by pixel into the FPGA and resulting in F, the final score computed for each pixel in the image.

There are 5 basic steps:

1. Compute ix and iy derivatives for each pixel in the image.
2. Queue five lines of the intermediate results ix*ix, iy*iy and ix*iy as A, B and C respectively.
3. Weigh a 5x5 window of A, B and C by an equidistant Gaussian distribution and sum the window together.
4. Compute the determinant and trace of M to find the score, F = Det – K* Tr*Tr. K is empirically chosen to be $0.04 < K < 0.15$ in literature [10] and for this implementation is 0.05.

5. Apply a threshold to F, so that if F is above a threshold then that pixel location is considered a valid feature. Note the coordinate of the valid feature.

The threshold is modified during run time to provide the correct quantity of features in the scene. This will change based on the contrast and texture available in the scene. Too low a threshold will provide too many features to be processed in time. Too high a threshold provides too few features for good accuracy in the matching algorithms.

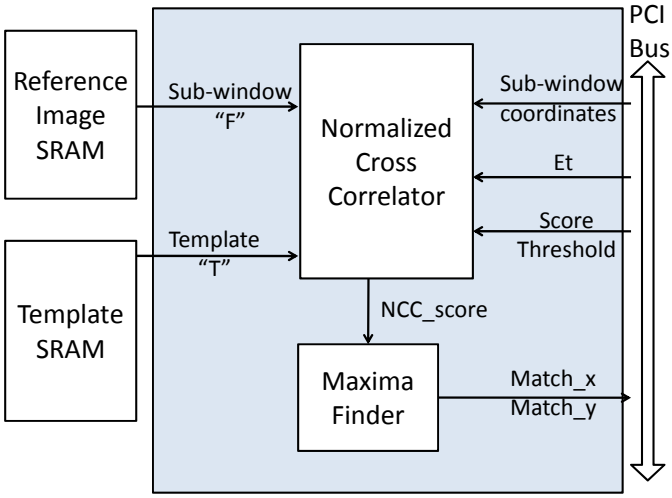
After a fixed delay to fill the internal queues, a score will be produced after each successive clock. Failing scores are discarded**.** Scores that are higher than the predetermined threshold are recorded to SRAM.

7. NORMALIZED CROSS CORRELATOR

The Normalized Cross Correlator is based almost exclusively upon the design by Zhang and Asari[12]. Our implementation is a re-creation of the work described in that paper.

The only customization is in the integration with the VISINAV system, in the interface to the memory banks, and in adding a maxima finder to report the highest score and its coordinates rather than every score across the image.

As the core design of the Normalized Cross Correlator is unchanged from that of Zhang and Asari[12], only the custom interface and control logic will be discussed here.



Sequence:

1. In software, load a 15x15 window from the rectified descent image around each feature that will be matched. Sum the intensity values across the 15x15 window and save as scalar EnergyT. Subtract the DC component over the window and convert to the log2 domain. Call the resulting 15x15 window "T".

2. In software, determine the sub-window of the map image or the prior descent image that will be searched for a match. Call the coordinates of this subwindow "wCoord", and the sub-window itself the "reference image".

3. Load the window T, the scalar EnergyT, and the sub-window coordinates wCoord to the FPGA.

4. In the FPGA, load the sub-window "F" from the reference image from on board SRAM using wCoord and feed into the Normalized Cross Correlator. The output will be an NCC score.

5. Save the coordinates of the maximal NCC score. If no score falls above a threshold then report no match detected.

Although this is a fully functional system, there are several design limitations that reduce the benefit of the FPGA acceleration and unduly burden the CPU. In this design the CPU is forced to compute the average and the log2 conversion of each template before running a single correlation in the FPGA. The averaging and log2 conversion of the template could be done on the fly in the FPGA without significant additional resources. Also, additional logic could be added to run all of the correlations at once, each template correlated across its sub-window reference image, rather than a single template match at a time. This would improve performance and reduce CPU interrupts. These improvements will be included in the next version of the NCC.

8. FUTURE AND RELATED WORK

Much of the logic that applies to putting pin point landing vision algorithms into an FPGA also apply to putting rover navigation algorithms into an FPGA. JPL has a significant body of tested, functional vision algorithms implemented in an FPGA for stereo vision processing for obstacle avoidance and is in development on visual odometry for state estimation. [7]

The release of the Xilinx Virtex 5 Radiation hardened and space qualified XQR5VFX130 part is a major breakthrough for vision based FPGA design and will enable an array of computationally intensive tasks for space missions that would not have been realistic previously.

As a comparison, the Mars Exploration Rover and the Mars Science Laboratory both flew FPGA's for the camera control and image passing, but they were roughly 4 times smaller and were fully consumed by the task of doing PCI bus control and DMA between the on-board memory banks. More critically, prior mission FPGA's have typically been fuse-based for radiation protection, and the Virtex 5 is

SRAM based but still flight qualified. This allows for an in-flight reprogramming capability that JPL did not previously use for the camera control FPGA. This means that the same FPGA could handle acceleration of the VISINAV algorithm during EDL and then get reprogrammed to handle acceleration of stereo and visual odometry algorithms for rover navigation once on the ground.

Full integration with the VISINAV software system has not yet been completed on this task- the FPGA modules were written with full integration in mind, but end to end testing has not yet been done. Performance will need to be validated for both the speed and the behavior as compared against the software only version. Some differences in results are expected; significant testing is needed to prove it is within a tolerable bound of accuracy.

9. RESULTS

The resource requirements for the Homography, Feature Detection and Normalized Cross Correlator are largely independent of image size, features tracked or image throughput.

Homography and NCC resource usage are dependent only upon the kernel size used for the image computation. For Homography kernel size is a 2x2 window and for NCC it is a 15x15 window.

The Harris Detector uses slices to queue certain types of image data, so slices will change with image size. Roughly 10% of slice count scales linearly with image size.

The standard build options for 512x512 are as follows:

FPGA Resource usage	BRAM[3]	Slices[4]
Virtex 4LX160	288	67,584
Virtex 5FX130t	576	40,960
Homography	0	4,255
Harris Detector	0	8,426
Normalized Cross Correlator	47	11,838
Complete System (incl. PCI bus interface etc)	47	25,556

Throughput performance depends on image size and the features tracked. While on previous software-only implementations of VISINAV as few as 70 features have been tracked, with this FPGA enhancement the features tracked can be an order of magnitude higher.

[3] BRAM is 18Kbit

[4] Slice count is presented in the Virtex 4 Family's 4 input LUT reference frame, not the Virtex 5 Family's 6 input LUT reference frame for a 1-1 comparison

Estimated timing results (using a 32 bit, 33Mhz PCI bus and a 66MHz FPGA clock):

(1) Time to transfer a 512x512 image one way: 3.6 ms

(2) Time to perform Feature detection using Harris operator across descent image (512x512): 4.0ms

(3) Time to perform Homography across descent image (512x512): 4.0ms

(4) Time to perform Normalized Correlation between one 15x15 template and a 32x32 search window: 19us

(5) Time to perform Normalized Correlation across 500 features:9.5 ms

(6) Time to transfer 512x512 rectified image back: 3.6 ms

25 ms total per frame within the FPGA. Our 1Hz target throughput leaves 975 ms for the CPU to perform the rest of VISINAV including (d) Fast Fourier Transform and (e) Kalman filter state updates.

Timing testing indicates that anywhere from 1.72 seconds to 7.32 seconds could be spent on the Fast Fourier Transform depending on map image size (1500x420 and 3000x840 pixels respectively). The FFT is thus a required addition to future work. Once included, 1-5 Hz timing is anticipated.

Timing estimates depend upon the FPGA place and route being able to meet a 66MHz clock requirement. 66MHz is a conservative clock rate- max clock rate for the current design is greater than 100MHz. Once the 66MHz clock frequency timing is met the only variation on the 25 ms per frame estimate is in the PCI transfer times. PCI transfer times were measured empirically from 33MHz PCI systems available to hand, and are conservative.

10. SUMMARY

Pin point landing in hazardous terrain can be made possible with the addition of vision algorithms such as VISINAV, and the use of FPGA components can reduce the computational load on the CPU enough to enable real flight missions.

The resource requirements of Homography, Feature Detection and Correlation are within the limits of the expected flight chip, the Virtex 5 FX130. The throughput expected with the use of FPGA enhancement is in the 1-5Hz range for Feature Tracking and 0.5 to 1Hz for Mapped Landmark. Mapped Landmark would require the addition of a FFT FPGA component to reach the 1-5Hz range.

The existence of similar FPGA modules being developed at JPL to handle rover navigation once on the ground indicate that the same Virtex 5 FPGA could serve multiple functions during difference phases of the same mission. A common architecture for RAM and PCI interfaces exists to increase

the code re-usability between the VISINAV FPGA code and the rover navigation FPGA code.

ACKNOWLEDGEMENTS

The work described in this publication was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract from the National Aeronautics and Space Administration. This work was developed and matured under internal R&D programs.

The authors would like to thank Andrew Johnson, Larry Matthies, Adnan Ansar, Anastasios Mourikis, Nikolas Trawny, and Stergios I. Roumeliotis for their work on designing the VISINAV algorithm, the work from which all of the FPGA design is derived. The authors would also like to thank Steve Goldberg for his advice and support on software integration.

REFERENCES

[1] NASA, "Solar System exploration roadmap", http://solarsystem.nasa.gov/multimedia/downloads/SSE_RoadMap_2006_Report_FC-A_opt.pdf.

[2] A. E. Johnson, A. Ansar, L. H. Matthies, N. Trawny, A. I. Mourikis, and S. I. Roumeliotis, "A general approach to terrain relative navigation for planetary landing," AIAA Aerospace@Infotech Conf., Rohnert Park, CA, May 2007.

[3] N. Trawny, A. Mourikis, S. Roumeliotis, A. Johnson, J. Montgomery,A. Ansar, and L. Matthies, "Coupled vision and inertial navigation for pin-point landing," presented at the NASA Sci. Technol. Conf. (NSTC2007), College Park, MD, Jun. 19–21.

[4] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, and L. Matthies. "Vision-Aided Inertial Navigation for Precise Planetary Landing: Analysis and Experiments" In Proceedings of Robotics: Science and Systems, Atlanta, GA, June 26-30, 2007.

[5]Roumeliotis, S., Johnson, A., and Montgomery, J. 2002. "Augmenting inertial navigation with image-based motion estimation". In Proc. Int'l Conf. Robotics and Automation (ICRA 2002), pp. 4326- 4333.

[6]C. Seybold, G. Chen, P. Bellutta, A. Johnson, L. Matthies, and S. Thurman, "Suborbital flight test of a prototype terrain-relative navigation system," AIAA Infotech at Aerospace Conference, Rohnert Park, CA, May 2007.

[7]C. Villalpando, A. Morfopolous, S. Goldberg, L. Matthies, "FPGA Implementation of Stereo Disparity with High Throughput for Mobility Applications", IEEE Aerospace Conference, March 5-12, 2011, Big Sky, Montana. (expected)

[8]R. Tsai, and T. Huang, "Estimating three-dimensional motion parameters of a rigid planar patch", IEEE ASSP Vol. 29, No 6, 1981.

[9] Tsai, R., T. Huang, and W. Zhu, "Estimating three dimensional motion parameters of a rigid planner patch, II: Singular Value Decomposition", IEEE ASSP, Vol. 30, No 4, 1982.

[10]Zhang, Z., R. Deriche, O. Faugeras, Q.T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," Artificial Intelligence, 78, (1995), pp. 87-119

[11]Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings, 4th Alvey Vision Conference. (1988) 147–151 Manchester.

[12]Ming Z. Zhang and K. Vijayan Asari, "A hardware-efficient high performance digital architecture for run-time computation of normalized cross correlation," WSEAS Transactions on Circuits and Systems, vol. 5, no. 9, pp. 1416-1423, September 2006.

BIOGRAPHY

Arin Morfopoulos *is a member of the Robotic Actuation and Sensing Group at the Jet Propulsion Laboratory. He has been active in the FPGA design and implementation of vision algorithms on a half dozen DoD and NASA projects. He has been responsible for the system interfaces and integration of the FPGA vision algorithms on those tasks since 2007. He received a B.S. in Electrical Engineering from UCLA.*

Brandon Metz *is a member of the Robotic Actuation and Sensing Group at the Jet Propulsion Laboratory. At JPL he has worked in FPGA design, analog circuit design, system level design and is currently working on hardware testing for Mars Science Laboratory. He earned his B.S. in Electrical Engineering and his M.S in Communication and Microwave Engineering from Cal Poly Pomona.*

Carlos Y. Villalpando *is a Senior Member of Technical staff in the Advanced Computer Systems and Technologies group at the Jet Propulsion Laboratory. He is currently a digital designer for advanced computing techniques for machine vision applications in FPGAs as well as system designer and programmer for machine vision tasks on multicore processors. He earned his Bachelor of Science degree in Electrical Engineering, Computer Block at the University of Texas at Austin in 1996 and a Master of Science in Electrical Engineering-VLSI at the University of Southern California in 2003. He has been a member of the JPL community continuously since 1993 and has worked primarily on Technology development tasks.*

Dr. Larry Matthies *obtained a Ph.D. degree in Computer Science from Carnegie Mellon University in 1989, then moved to the Jet Propulsion Laboratory, where he is currently a Principal Member of Technical Staff and supervisor of the Machine Vision Group. His research has focused on terrain sensing and obstacle avoidance algorithms for autonomous navigation of robotic vehicles. At JPL, he pioneered the development of real-time algorithms for stereo vision-based obstacle detection and he contributed to the development of the structured light sensor used by the Sojourner Mars rover. He has also developed algorithms for visual motion estimation from image sequences, 3-D scene reconstruction from image sequences, real-time terrain classification using multispectral imagers, and environmental mapping using sonar and stereo vision sensors. His group currently has research projects on computer vision for robotic vehicles sponsored by NASA, DARPA, and the U.S. Army; these projects include work on navigation of Mars rovers, asteroid and comet landers, and Earth-based robotic vehicles for urban and cross- country missions. He is a member of the editorial board of the Autonomous Robots journal and an adjunct member of the Computer Science Department at the University of Southern California. He has also been an invited speaker at the Frontiers of Engineering Symposium organized by the National Academy of Engineering.*

Navid Serrano *was a member of the Computer Vision group at the Jet Propulsion Laboratory, Pasadena, CA, during the work described in this paper. Prior to joining JPL he was a member of the research staff at Eastman Kodak Company, Rochester, NY, where he worked in the areas of image understanding and image enhancement. His focus is on hazard detection and analysis in orbital data for landing site selection.*